

DroidRide: Buckle up Security Belt When Enjoying Ride in Android World

SYSCAN 2012

Wayne Yan

Difference between Oct. and Dec



*Obama: thank
your for
postponing
SYSCAN to wait
for my re-election!*



Why is DroidRide (Joyride)?

- Roxette: "Come on, join the **droidride** everybody,
- get your tickets here, step right this way"



About me

■ automaton fan

- If there is an automatic approach, I never do it manually.
- Always look for interested security projects

■ Network-level

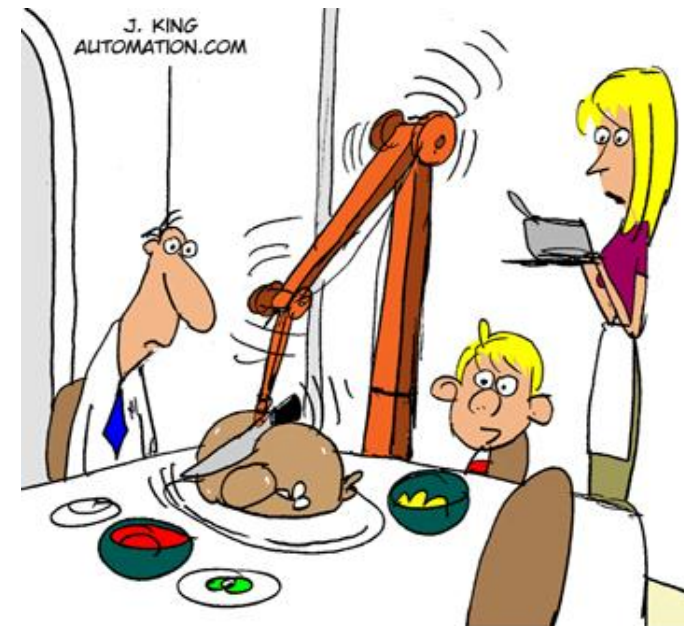
- 10Gbps+ Stream-based next generation firewall design
- Million-scale botnet sample analysis

■ Desktop-level

- Malware automatic processing
- Cloud-based AV
- 20+ Packers
- NTFS raw hard disk, Anti-rootkits

■ Scalable anti-malware engine for android

Speaker: Li-Ming Chen



“The control system is based on my CARVE algorithm...Cut And Render, Verily Eat.”

Fundamental questions?

- Would cloud-based anti-virus still appear on the market if the virus signature file was less than 100MB in 2007?
 - **NO** IMHO
- If you don't need desktop-based anti-malware solution, is the “cloud garbage bin” the first thing in your mind?
 - **Well... maybe** Signatures, MD5, URL, Sandbox, etc.,
 - Thin desktop and “garbage cloud”
- Smarter and lightweight engine to defend mass-production malware, especially for the new android world? **YES!**



Disclaimer

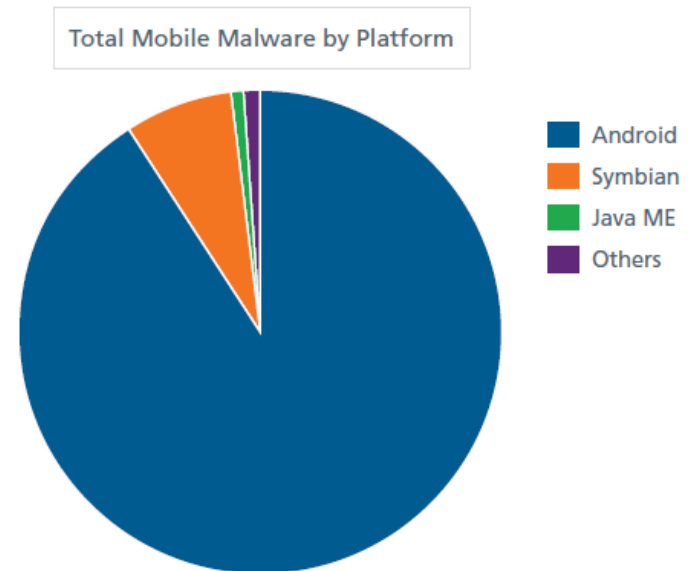
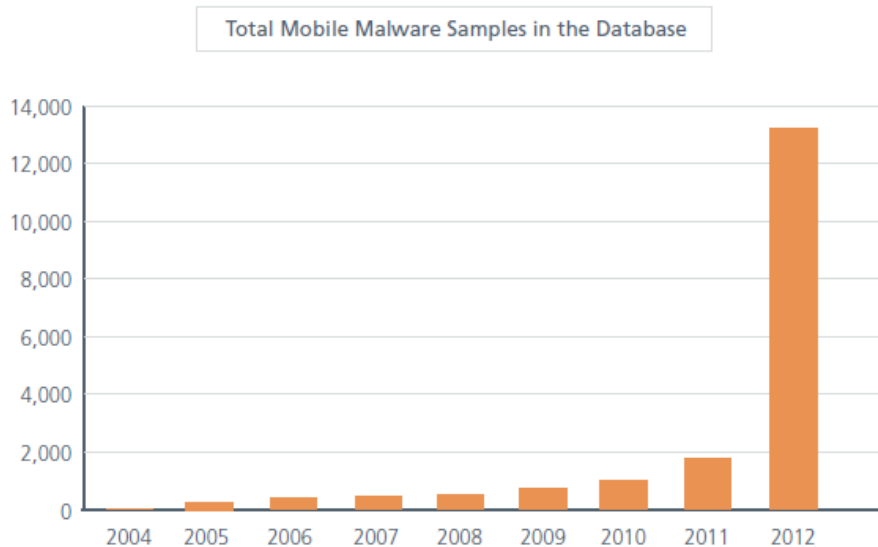
- Individual research work
- Does not represent, showcase or endorse related products of any of my present and previous employers
- No marketing here

Agenda

- Current android threat landscape
- Differences between PC and mobile-based solutions
- Droidride engine
- Demo
- Q&A

Android threat landscape

- McAfee's Q2 report: Android OS is the most popular target for writers of mobile malware.
- How many android malware right now? (correct me if wrong)
 - Wild guess: 300k ?
 - Whitelisting: 1M (some android apps from China not included) ?



PC malware years ago- where?

Number of new samples added to AV-Test.org's malware database (virus collection)

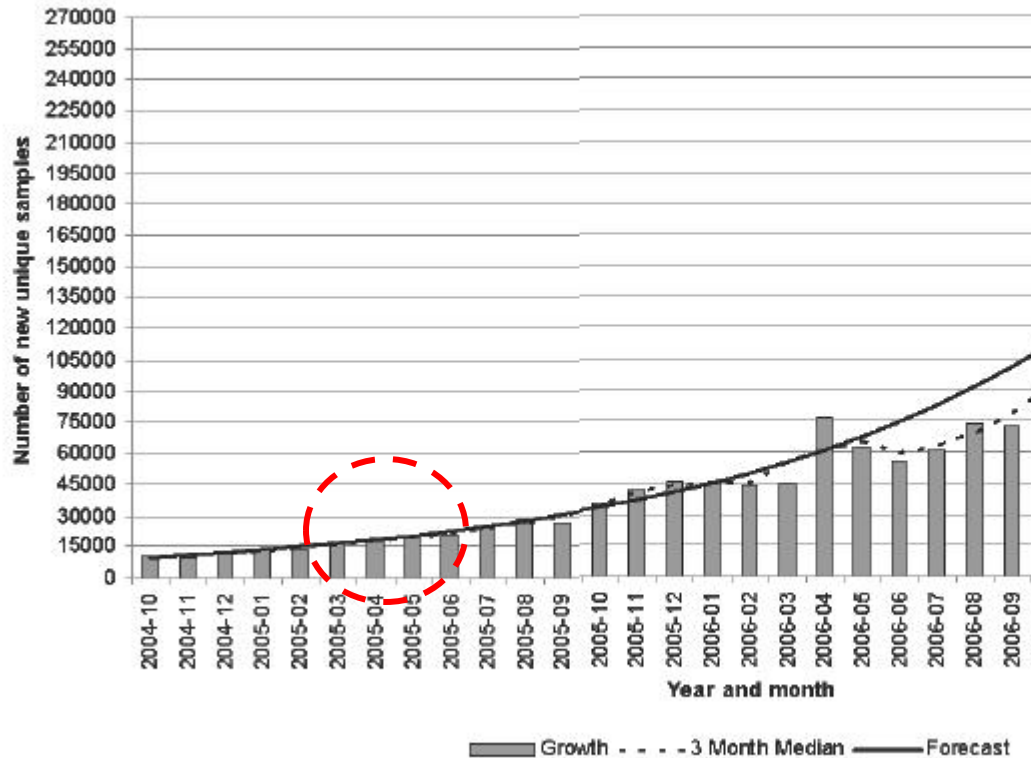
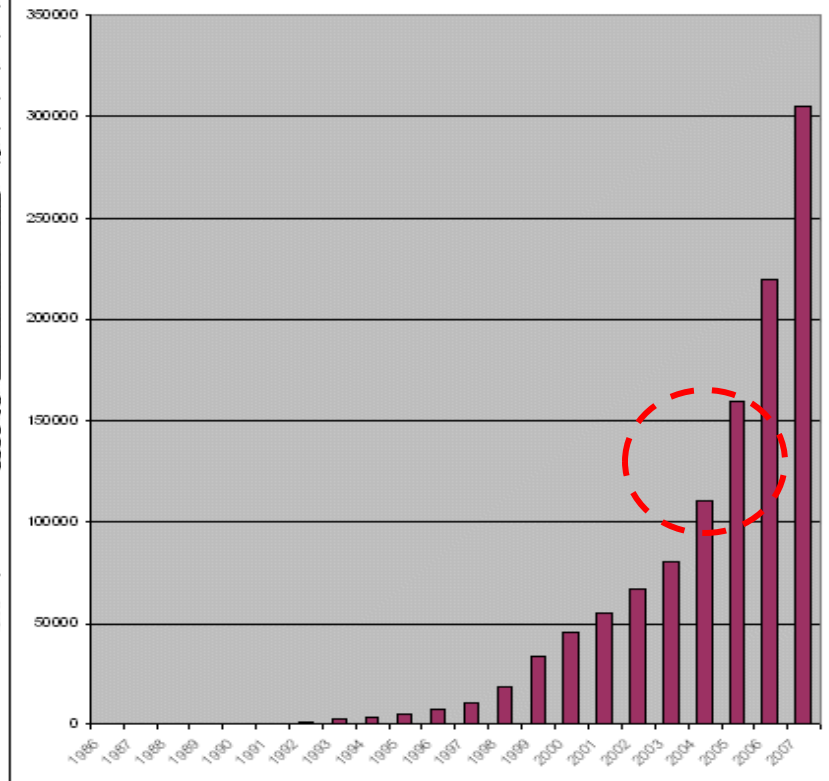


Figure 2: AV-Test's malware collection growth.

Number of viruses 1986-2007





What happened in PC anti-malware after 2007

?

AV companies cannot get hold of new virus and variants.

Detection rates dropped from 95+% to as low as 79%

Clients started to complain about high CPU usages, and computer were getting slow down...

You spent 40 bucks on anti-virus software and didn't catch any piece of malware in a whole year (free but blocking every software is a different story 😊)

Gap between frontend product lines and backend processing

Startups appeared on the market for APT profit margins

PC anti-malware vs. mobile anti-malware



- Are you busy cloning anti-malware solutions from pc to mobile?
- Unfortunately, copycat doesn't work!
 - Out of 41 different Android anti-virus solutions tested in 2012, AV-Test found that 34 scanners detected less than 65% of 618 malware.



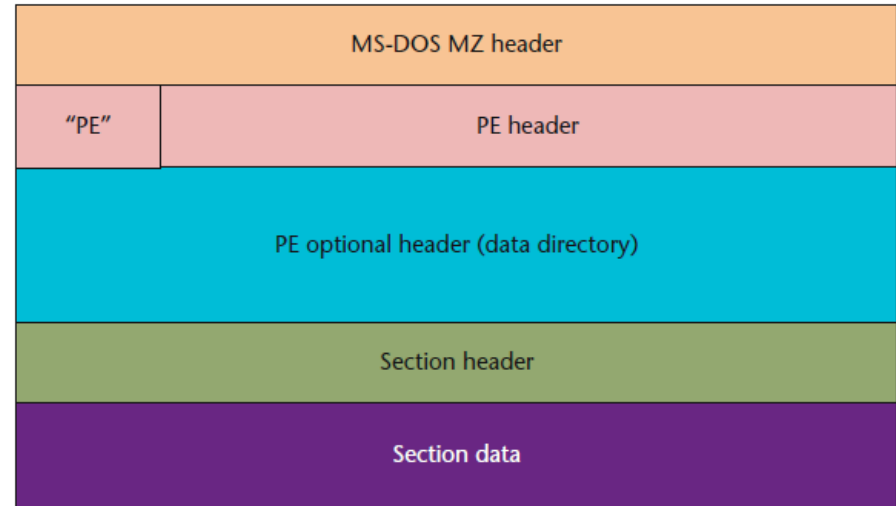
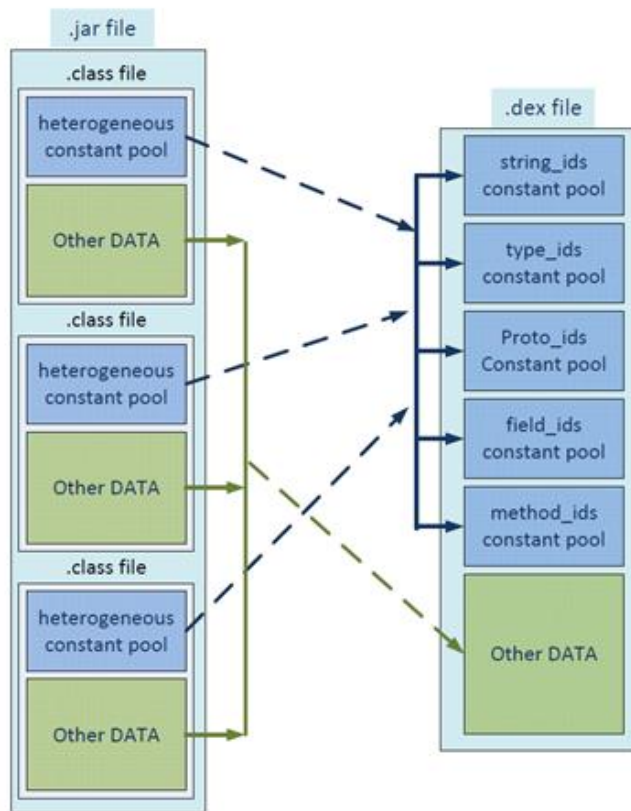
Things are getting worse in mobile devices

- Battery constrains
- Limited CPU computations, not suitable for behavior-based emulation
- Internet bandwidth
- Current products can be easily fooled by simple obfuscations
- Wireless carrier networks are quite different from wired ones

PE vs. DEX 101

■ Windows PE format

- ❑ Data
- ❑ Resources
- ❑ Strings
- ❑ API functions



■ Android dex

- ❑ Strings
- ❑ Codes
- ❑ Rigid formats
- ❑ Work with other files, such as resources and permission file

Mobile rootkit (not a big issue though)

PE packer vs. embedded dex

More and more android packers

```
<!DOCTYPE html>
<html>
<head prefix="og: http://ogp.me/ns# fb: http://ogp.me/ns/fb# githubog: http://ogp.me/ns/fb/githubog#"
<meta charset='utf-8'>
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<title>APKfuscator/resources/apkcrypt.dex at master · strazzere/APKfuscator · GitHub</title>
<link rel="search" type="application/opensearchdescription+xml" href="/opensearch.xml" title="GitHub" />
<link rel="fluid-icon" href="https://github.com/fluidicon.png" title="GitHub" />
<link rel="apple-touch-icon-precomposed" sizes="57x57" href="apple-touch-icon-114.png" />
<link rel="apple-touch-icon-precomposed" sizes="114x114" href="apple-touch-icon-114.png" />
<link rel="apple-touch-icon-precomposed" sizes="72x72" href="apple-touch-icon-144.png" />
<link rel="apple-touch-icon-precomposed" sizes="144x144" href="apple-touch-icon-144.png" />

<link rel="icon" type="image/x-icon" href="/favicon.png" />

<meta content="authenticity_token" name="csrf-param" />
<meta content="yX38ZHsEDbgd87NEep7WgZz6/hTb7ooqiCRfnSmzVUI=" name="csrf-token" />

<link href="https://a248.e.akamai.net/assets.github.com/assets/github-324e543186da6868cee3c68ba953e4a4" />
<link href="https://a248.e.akamai.net/assets.github.com/assets/github2-a833fa188e668e3c5b7dec9668a7605
```

late Results - DEXTemplate.bt				
Name	Value	Start	Size	Cc
t odexpad	0	(local)		
t odex	0	(local)		
char tmp[3]		(local)		
- char tmp[0]	32 ''	(local)		
- char tmp[1]	32 ''	(local)		
- char tmp[2]	10	(local)		
struct header_item dex_header		0h	8h	Fg:
struct dex_magic_magic	<!	0h	8h	Fa:

Startup classes.dex															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
0000h:	64	65	78	0A	30	33	35	00	4C	70	4F	08	FA	8A	70
0010h:	38	2A	18	09	AB	09	91	BD	D1	18	65	4E	3B	40	B9
0020h:	FC	8D	01	00	70	00	00	00	78	56	34	12	00	00	00
0030h:	00	00	00	00	2C	8D	01	00	72	04	00	00	70	00	00
0040h:	1F	01	00	00	38	12	00	00	0F	01	00	00	B4	16	00
0050h:	82	01	00	00	68	23	00	00	99	03	00	00	78	2F	00
0060h:	67	00	00	00	40	4C	00	00	DC	34	01	00	20	59	00
0070h:	E4	49	01	00	A2	34	01	00	8B	6B	01	00	5E	5E	01
0080h:	5F	41	01	00	DF	3A	01	00	E5	3A	01	00	07	3A	01
0090h:	39	5F	01	00	31	5F	01	00	79	6A	01	00	3B	63	01
00A0h:	C9	31	01	00	00	4C	01	00	A8	6A	01	00	61	65	01
00B0h:	2D	46	01	00	BF	62	01	00	21	63	01	00	50	65	01
00C0h:	38	58	01	00	00	54	01	00	2E	66	01	00	CF	49	01
00D0h:	72	3F	01	00	17	33	01	00	4F	33	01	00	D8	39	01
00E0h:	C2	39	01	00	67	40	01	00	74	39	01	00	50	2F	01
00F0h:	96	54	01	00	C5	59	01	00	77	5A	01	00	7C	5A	01
0100h:	AC	62	01	00	DF	68	01	00	CB	59	01	00	5B	66	01
0110h:	B5	59	01	00	BB	59	01	00	E3	6F	01	00	06	47	01
0120h:	D3	35	01	00	E9	31	01	00	3C	32	01	00	F3	3C	01
0130h:	82	28	01	00	6C	6A	01	00	C9	40	01	00	BF	40	01
0140h:	E8	35	01	00	E3	47	01	00	DF	47	01	00	F9	62	01
0150h:	8B	65	01	00	D4	62	01	00	B7	62	01	00	F2	62	01
0160h:	83	65	01	00	58	46	01	00	B7	70	01	00	A3	70	01
0170h:	8A	70	01	00	E0	70	01	00	CF	70	01	00	79	70	01
0180h:	09	71	01	00	F5	70	01	00	10	4E	01	00	DF	4E	01
0190h:	03	4F	01	00	0A	49	01	00	9C	6D	01	00	80	6E	01
01A0h:	37	72	01	00	5E	46	01	00	C2	67	01	00	C1	53	01
Template Results - DEXTemplate.bt															
Name		Value													
int odexpad		0													
int odex		0													
char tmp[3]		dex													
struct header_item dex_header															
struct string_id_list dex_string_ids		1138 strings													
struct type_id_list dex_type_ids		287 types													
struct proto_id_list dex_proto_ids		271 prototypes													
struct field_id_list dex_field_ids		386 fields													
struct method_id_list dex_method_ids		921 methods													

Malicious payloads will be hidden in files other than dex files

Android packer APKfuscator (Strazzere)

- 0-255
- 263-1917
- 1925,4073
- 4081,10188
- 10196,11295
- 11327,11393
- 11513,11588
- 11596,12120
- 12128,12225
- 12233,12465
- 12508,12917
- 12925,13031
- 13087,13465
- 13471,13643
- 13651,13845
- 13853,14129
- 14137,17508
- 17510,29163

```

2020 3C6D 6574 6120 6874 7470 2D65      <meta http-e
6976 3D22 582D 5541 2D43 6F6D 7061      quiv="X-UA-Compa
626C 6522 2063 6F6E 7465 6E74 3D22      tible" content="
3D65 6467 6522 3E0A 2020 2020 2020      IE=edge">.
3C74 6974 6C65 3E41 504B 6675 7363      <title>APKfusco
6F72 2F72 6573 6F75 7263 6573 2F61      ator/resources/a
6372 7970 742E 6465 7820 6174 206D      pkcrypt.dex at m
7465 7220 C2B7 2073 7472 617A 7A65      aster .. strazze
2F41 504B 6675 7363 6174 6F72 20C2      re/APKfuscator .
4769 7448 7562 3C2F 7469 746C 653E      . GitHub</title>
2020 203C 6C69 6E6B 2072 656C 3D22      . <link rel="
6172 6368 2220 7479 7065 3D22 6170      search" type="ap
6963 6174 696F 6E2F 6F70 656E 7365      plication/opens
6368 6465 7363 7269 7074 696F 6E2B      archdescription+
6C22 2068 7265 663D 222F 6F70 656E      xml" href="/open
6170 6269 6E73 6D6C 2220 7469 746C      ---h---" title
me-classes.dex
0A0A 0A3C 2144 4F43 5459 5045 2068      ...<!DOCTYPE h
6C3E 0A3C 6874 6D6C 3E0A 2020 3C68      tml>.<html>. <h
6420 7072 6566 6978 3D22 6F67 3A20      ead prefix="og:
7470 3A2F 2F6F 6770 2E6D 652F 6E73      http://ogp.me/ns
6662 3A20 6874 7470 3A2F 2F6F 6770      # fb: http://ogp
652F 6E73 2F66 6223 2067 6974 6875      .me/ns/fb# githu
673A 2068 7474 703A 2F2F 6F67 702E      bog: http://ogp.
2F6E 732F 6662 2F67 6974 6875 626F      me/ns/fb/githubo
223F 0A20 2020 203C 6D65 7461 2063      s#> <meta c

```

Applications: close-loop vs. open-loop

■ Close-loop

- ❑ Marketing cost is higher when not choose official app stores, e.g. iOS

■ Semi-Open-loop

- ❑ Third-party Android app stores in China
- ❑ Android store is by-passed in China, sorry!
- ❑ free and ad-based apps

exploit vs. defend

- Why huge achievements in spacecraft (it looks like more advanced than chip)? But still need to import jet engines.

Not even to mention the joke of “HanXin”



- Cannot control the defect protection of pipeline
- But you can concentrate the resources on making a superior one regardless of the fact of tens of defected ones
- Same story as



and “小学取消长跑和铅球”

Hackers vs. Anti-malware engine

- How hackers look like? Smart + fancy toolkits



- AV engine
 - Newbie
 - + robust system



File-based or stream-based anti-malware?

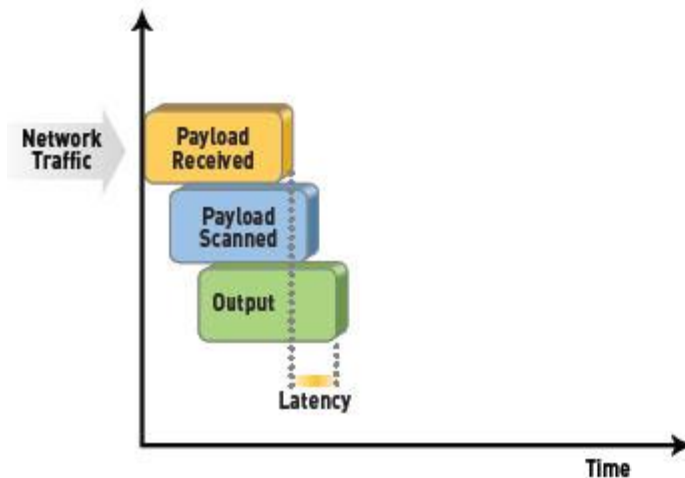
- Facts:
- PANS may take Yahoo! Space



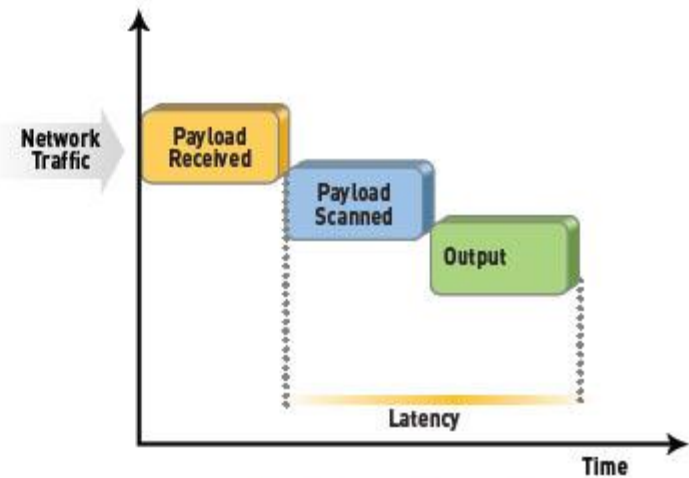
- Hash-based or more intelligent?
- Emulation is so expensive
- Big security vendors started to loose APT market shares to security startups
- Zero-day threats

Pics from google.com

What is stream-based ?

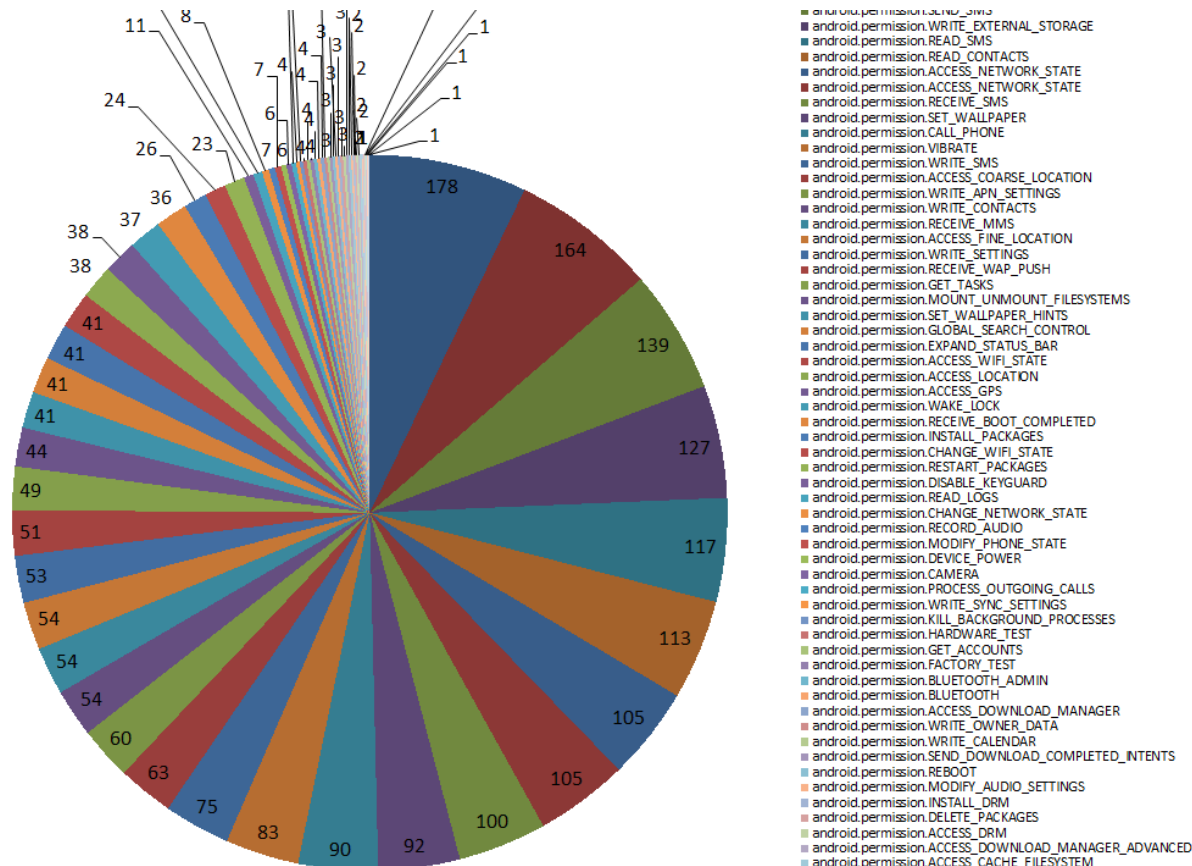


Stream-based Scanning



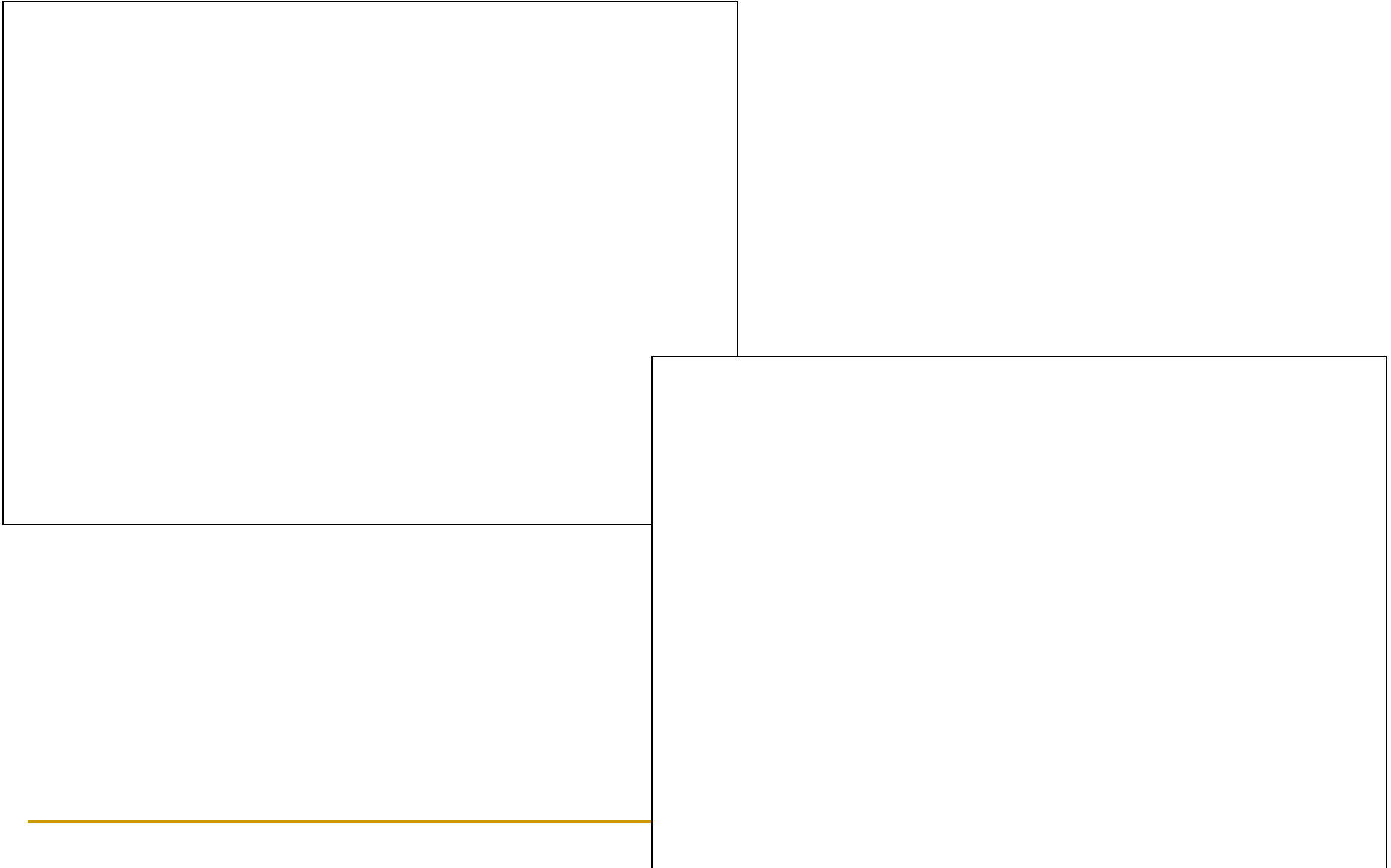
File-based Scanning

What we have learnt from PC anti-virus?



Lost in the API calls?

Current mobile anti-malware market



Current mobile anti-malware market

- 40+ claimed android anti-malware products
- Scan 50-100 installed apps on mobile devices
- Anti-malware testing standard for android malware is not ready
- Most by leveraging android open source SDK and toolkits
- A few numbers: 2001 → 1MB 2005 → 10MB
- 2008 → 100MB, 35% I will explain them.

Mobile anti-malware products between China and US

- Function menu
 - Traffic monitoring
 - SMS blocking
 - Widgets
 - Clients and market
 - Business model and profit margin
 - BYOD MDM MAM
-
- And more...

Cloud-based solution

- Current no clear picture of mobile anti-malware in the cloud
- Most works on backend-style malware scanning/emulation
- Signature not compatible with high-speed networking devices
- Call for new technologies

Engine design requirements

- Lightweight
 - Less CPU usage
 - Scalable to 1M – 10M android malware in 1-3 years

 - online detection (stream-base is a plus)
 - mobile devices and gateway
 - Larger overlapping between frontend and backend

 - Malware family correlation
 - Detect the whole malware family
 - Zero-day

 - Fully-automaton
-

Sample processing

APK parser

Manifest, dex parser

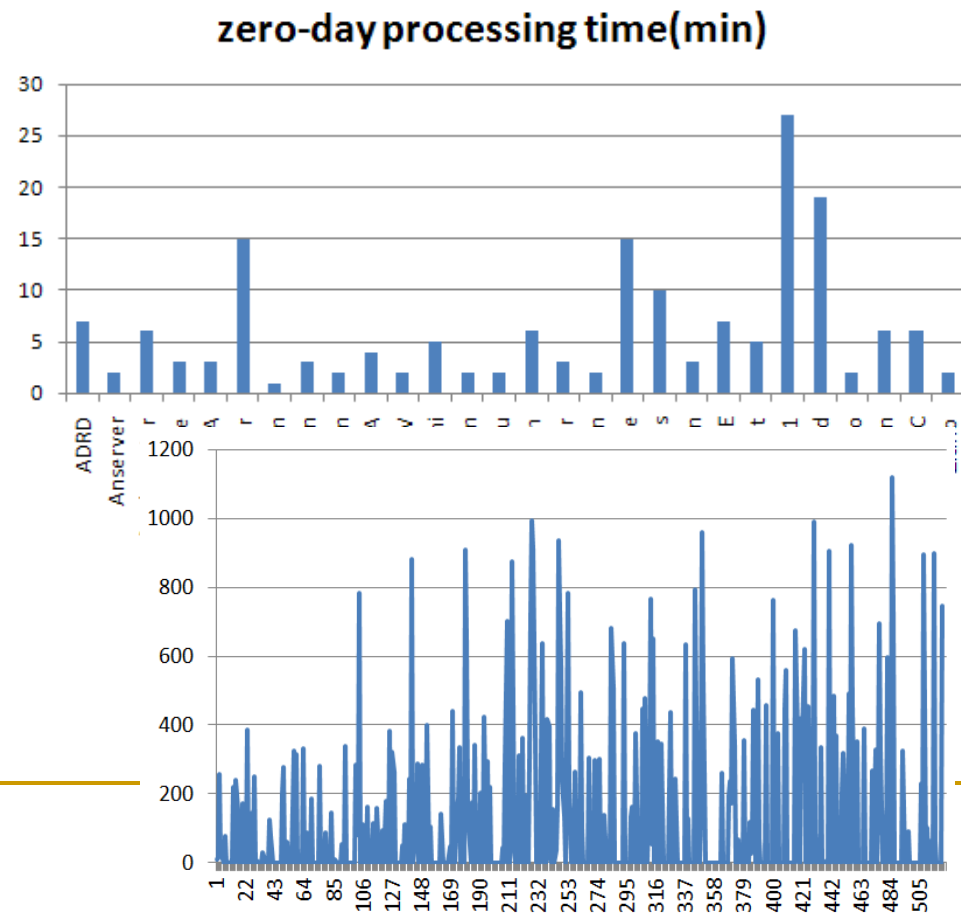
Statistic similarities

disassembly

Family features

Family correlations

- Inter-family correlation
- Detect new variants
- Weight score
- fingerprint distribution



Inter-family correlation

Root_exploit **ddream**
ESCOVPRIV

PJAPPS ADRD
KongTouTou

walkinwat

kmin opfake
infostealth

GinMaster	Feature range	index
	3650-3800	d
	3800-3980	d
	200-52000	10
	800-52000	10
	900-15000	10
	76000	3
	45800	3
	5c000	3

airpush

e55	13
428b	13
381	13
e73, 13f7	11
5231,7bd	11
5fb	13
e44f	5
d0d3	5
8dd4	5
1407	3
400,407	3

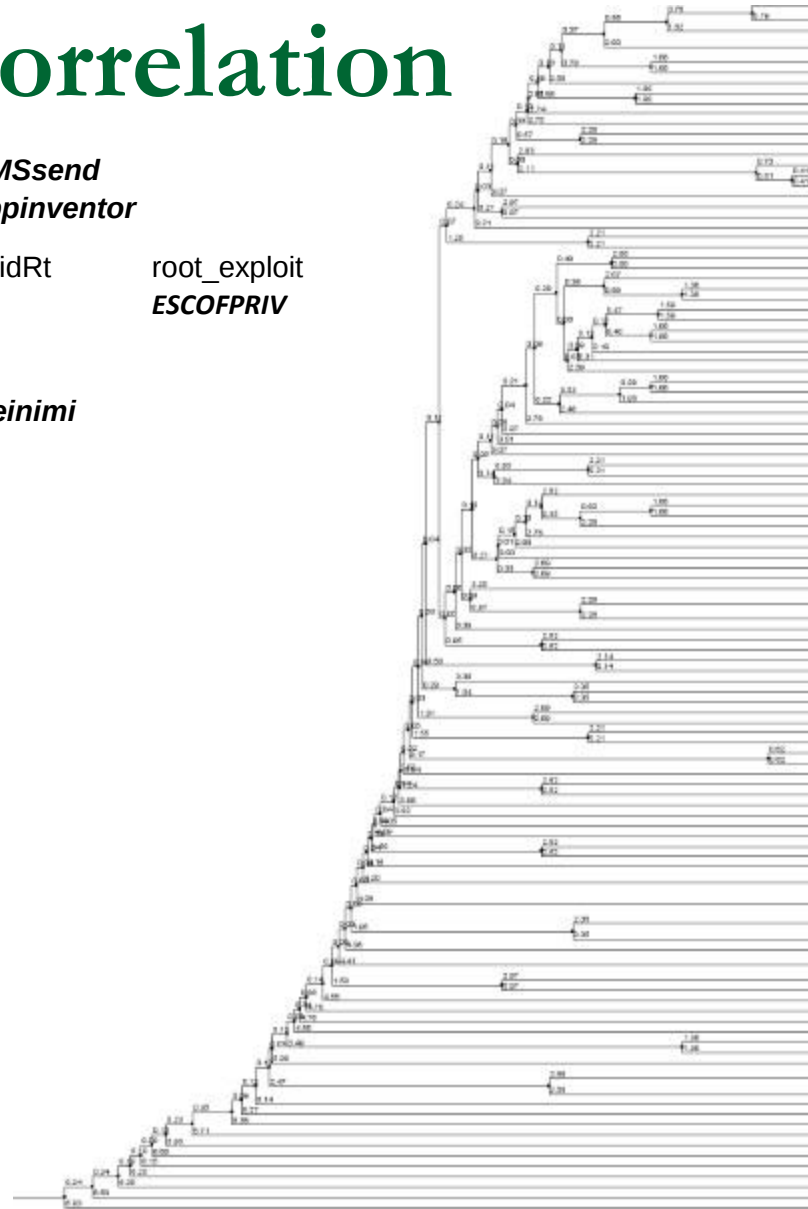
zsone

7103,7ebb	3
8200-8500	3
8b00-8f00	3

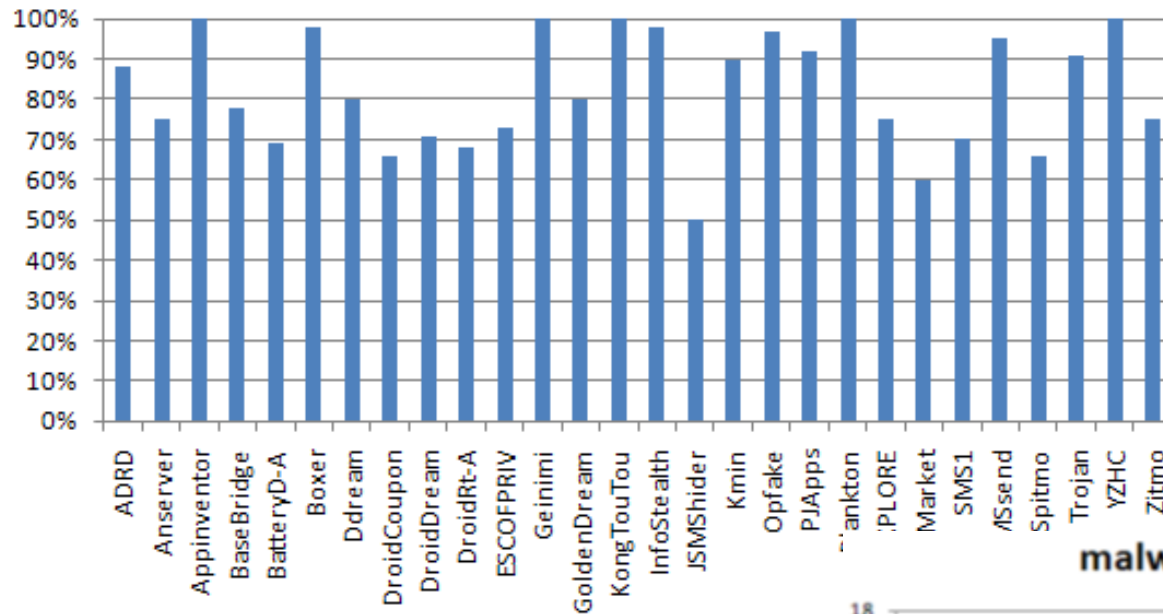
SMSsend
appinventor

droidRt root_exploit
ESCOFPRIV

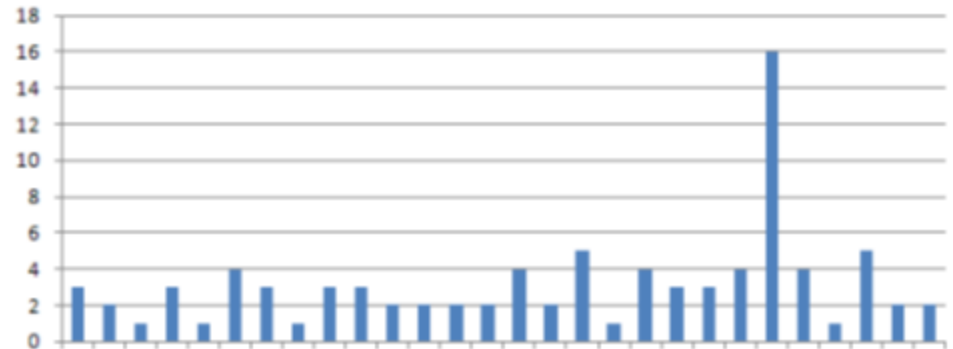
geinimi



detection rate



malware signature number

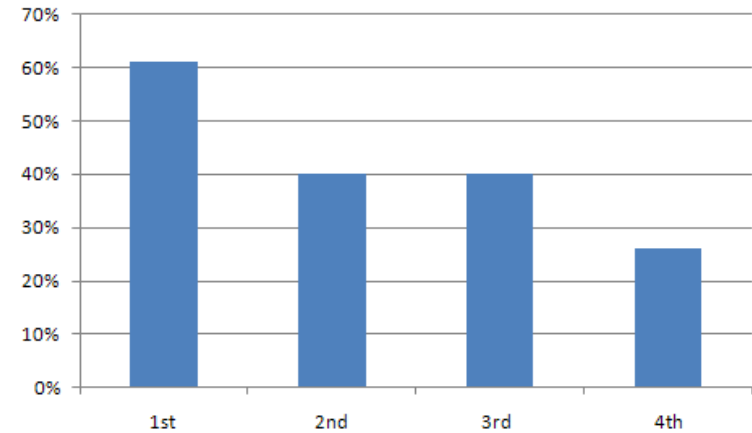
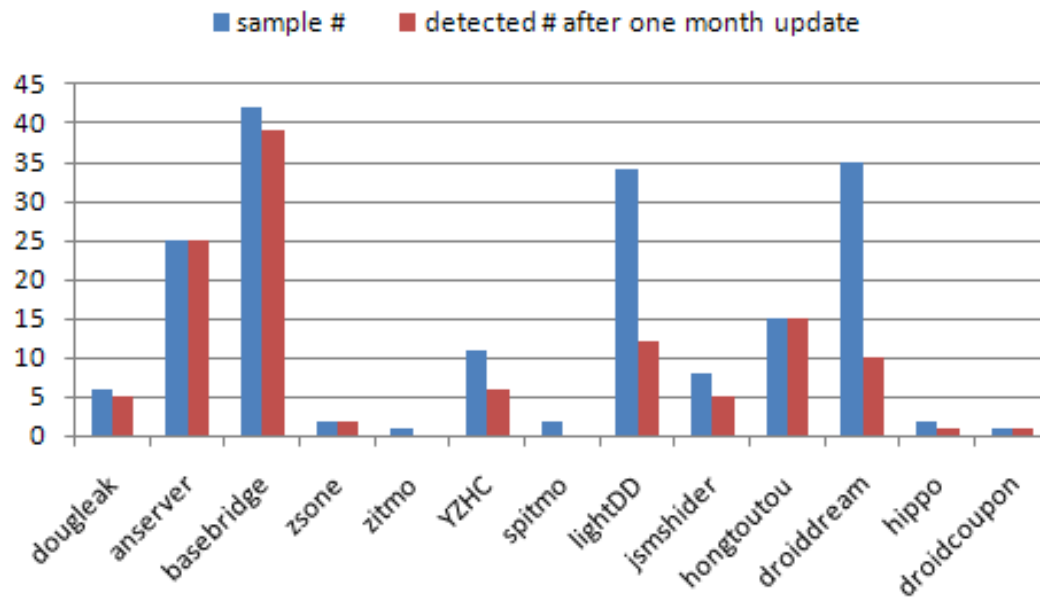


false positive filtering

- Malware family correlation
- Whitelisting filtering
- Signature confidence score
- Various logic operations
- Collected 10k Apps, 1 fp

Zero-day detection

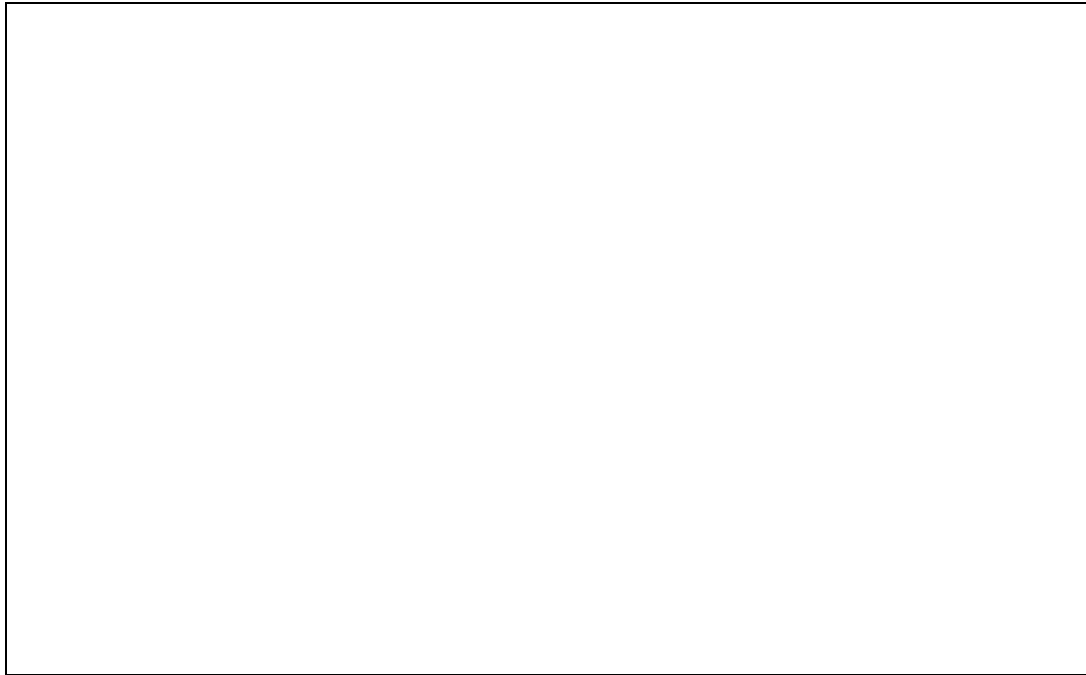
- New malware from malware families w/o updating signatures
- Mixed and unknown samples



- Keep signature out of date for one month and detect new samples
- Detection rate for 4-month window period w/o updating signatures

Zero-day detection cont'd

- How sample collections affect detection rate?
 - # of engines detected



Anti-obfuscation – 7 tricks

■ Alignment

- ❑ Optimize apk files – zipalign
- ❑ changes hash values

■ Re-sign

- ❑ generate a different signature for apk file, changed hash

■ Rebuild

- ❑ disassembles into Smali codes and rebuild assembly codes back into apk file;
- ❑ different Dalvik bytecode order but keep the original logic
- ❑ Changed hash, internal offsets, sizes

Reference: research work by M.Zheng,et al. DIMVA 2012

Anti-obfuscation cont'd

■ Inserting dummy methods

- Changed method names, order, method table

■ Renaming methods

- Defeat method-name based hash detection methods

■ Changing control flow graphs

- Insert go-to statements, changed CFG, defeat CFG-based signatures

■ Encrypting constant strings

- changed string and method tables, changed lots of stuff, not only strings

Anti-obfuscation testing

testing results by M.Zheng,et al.

AV Products	Original	Alignment	Re-sign	Rebuild
Kaspersky	95.95%	94.34%	94.59%	94.76%
F-Secure	95.50%	95.75%	95.05%	91.90%
Emsisoft	94.59%	93.87%	93.69%	75.24%
Ikarus	94.59%	94.34%	93.69%	75.24%
GData	94.14%	93.87%	93.69%	90.95%
TrendMicro	94.14%	91.98%	92.79%	77.62%
NOD32	92.79%	88.68%	88.29%	95.24%
Sophos	92.79%	94.81%	94.14%	78.10%
Antiy-AVL	92.34%	91.98%	89.19%	72.38%
Fortinet	90.99%	89.15%	88.74%	71.43%
Overall Average	93.78%	92.88%	92.39%	82.29%

Out testing results:

alignmeng	re-sign	rebuild
100%	100%	100%

Anti-obfuscation cont'd

AV Products	Insert	Rename	Change CFG	Str. Encrypt
Kaspersky	93.81%	73.33%	94.76%	90.95%
F-Secure	90.00%	90.00%	90.48%	68.57%
Emsisoft	83.81%	26.67%	82.86%	25.24%
Ikarus	83.81%	26.67%	83.33%	25.24%
GData	90.95%	90.48%	91.43%	88.10%
TrendMicro	61.90%	61.90%	63.81%	35.71%
NOD32	95.24%	91.90%	95.24%	90.48%
Sophos	54.29%	54.29%	54.76%	49.05%
Fortinet	48.57%	15.71%	42.86%	16.67%
Overall Average	77.24%	55.00%	76.67%	50.95%

insert	rename	change CFG	Str. Encrypt
67%	67%	95%	50%

Str. Encrypt is a kind of simple packer for android apps.

demo

Some thoughts

- Current mobile malware
 - Light obfuscated
 - Short embedded URLs
- So called “mobile application risk system”
 - High fp
 - Performance bottleneck

Q&A

- Thanks !